```
 1 #include <iostream>
 2 #include <string>
 3 #include <algorithm>
 4
 5 int main() {
 6     std::string s("kulturuke");
 7     std::cout << s << std::endl;
 8     while( next_permutation(s.begin(),s.end()) )
 9         std::cout << s << std::endl;
10 }
```

what might happen if you try to compile, link and run this program?

§x.x, title

I get:

```
g++ -Wall scratch.cpp && ./a.out
kulturuke
kultuuekr
kultuuerk
... 21721 more
uuutrlekk
uuutrlkek
uuutrlkke
```

What if you sort the string before entering the do loop? I get 30240 lines:

```
g++ -Wall scratch.cpp && ./a.out
ekklrtuuu
ekklrutuu
ekklruutu
...
uuutrlekk
uuutrlkek
uuutrlkke
```

Is string a regular container?

```
1 #include <iostream>
2 #include <string>
3
4 int main() {
5     std::string s("Hello World!");
6     std::wstring ws(s);
7     std::cout << ws << std::endl;
8 }
```

This code does not compile. How to fix?

§x.x, title

I get errors on both line 6 and 7.

```
g++ -Wall scratch.cpp && ./a.out
scratch.cpp: In function 'int main()':
scratch.cpp:6: error: no matching function for call to
  'std::basic_string<wchar_t, std::char_traits<wchar_t>,
  std::allocator<wchar_t> >::basic_string(std::string&)'
...
scratch.cpp:7: error: no match for 'operator<<' in 'std::cout << ws'
...
```

Here is one way to "fix":

```
#include <iostream>
#include <string>

int main() {
    std::string s("Hello World!");
    std::wstring ws(s.begin(), s.end());
    std::wcout << ws << std::endl;
}
```

[p586],[p609]

```
 1 #include <iostream>
 2 #include <string>
 3
 4 int main() {
 5     std::string s1 = "Foo";
 6     std::string s2 = "Gaz";
 7     s2 = s1;
 8     s2[0] = 'B';
 9     std::cout << s1 << std::endl;
10 }
```

what might happen if you try to compile, link and run this program?

§x.x, title

I get:

```
g++ -Wall scratch.cpp && ./a.out
Foo
```

Just for curiosity, what if you try?

```
#include <iostream>
#include <string>

int main() {
    char s1[] = "Foo";
    char s2[] = "Gaz";
    s2 = s1;
    s2[0] = 'B';
    std::cout << s1 << std::endl;
}
```

Then I get:

```
g++ -Wall scratch.cpp && ./a.out
scratch.cpp: In function 'int main()':
scratch.cpp:7: error: ISO C++ forbids assignment of arrays
scratch.cpp:7: confused by earlier errors, bailing out
```

(same is true for C99, C89)

And of course, if you attempt 'char *' then you get a runtime error.

```
 1 #include <iostream>
 2 #include <string>
 3
 4 int main() {
 5     std::string s1 = "abbcccde";
 6     std::string::size_type p = s1.rfind("cc");
 7     s1.replace(p, 2, "XXX");
 8     std::string s2 = s1.substr(3, -2);
 9     std::cout << s2 << std::endl;
10 }
```

what might happen if you try to compile, link and run this program?

§x.x, title

I get:

```
g++ scratch.cpp && ./a.out
scratch.cpp: In function 'int main()':
scratch.cpp:8: warning: passing negative value '-0x0000000000000002'
  for argument 2 to 'std::basic_string<_CharT, _Traits, _Alloc>
  std::basic_string<_CharT, _Traits, _Alloc>::substr(typename _Alloc::size_type,
  typename _Alloc::size_type) const [with _CharT = char, _Traits
  = std::char_traits<char>, _Alloc = std::allocator<char>]'
cXXXde
```

What would the value of 'p' be if the substring "cc" was not found? std::string::npos which is 4294967295 on my machine.

```
1 #include <iostream>
2
3 int main() {
4     int a = 4;
5     int b = 2;
6     std::clog << a << b;
7 }
```

what might happen if you try to compile, link and run this program?

§x.x, title

I get:

42

What is the difference between cerr and clog?

Why was '≪' and '≫' chosen?

[p607] '≪' and '≫' was chosen also because they bind the right way, that is (cout ≪ a) ≪ b rather than cout ≪ (a ≪ b);

```
 1 #include <iostream>
 2
 3 struct A {
 4     virtual std::ostream & put(std::ostream &) const = 0;
 5 };
 6
 7 struct B : A {
 8     std::ostream & put(std::ostream & s) const { return s << 'B'; }
 9 };
10
11 std::ostream & operator<<(std::ostream & s, const A & a) {
12     return a.put(s);
13 }
14
15 int main() {
16     B b;
17     std::cout << b << std::endl;
18 }
```

what might happen if you try to compile, link and run this program?

§x.x, title

I get:

```
g++ scratch.cpp && ./a.out
B
```

[p612] This is a fine way to print out objects for which only a base class is known.

```
1 #include <iostream>
2
3 int main() {
4     double pi = 3.14159265358979323846;
5     std::cout << pi << std::endl;
6     std::cout.precision(3);
7     std::cout << pi << std::endl;
8 }
```

What does this print out? How could this code might look like if we were using a stream manipulator instead?

§x.x, title

I get:

```
g++ scratch.cpp && ./a.out
3.14159
3.14
```

You might consider using a manipulator instead. Eg,

```
#include <iostream>
#include <iomanip>

int main() {
    double pi = 3.14159265358979323846;
    std::cout << pi << std::endl;
    std::cout << std::setprecision(3) << pi << std::endl;
}
```

[21.4.6.2 Standard I/O Manipulators, p633]

```
 1 #include <fstream>
 2 #include <iostream>
 3 #include <string>
 4
 5 int main() {
 6
 7     std::string ostr = "This is a test of writing and reading from files";
 8     std::ofstream ofile("myfile.tmp");
 9     ofile << ostr;
10
11     std::string istr;
12     std::ifstream ifile("myfile.tmp");
13     ifile >> istr;
14
15     std::cout << istr;
16 }
```

What might happen if you try to compile, link and run this program?

§x.x, title

I get nothing, beacuse ofile is not flushed before we start reading from the file. Adding a ofile.flush() might help a bit. You can also do an explisit ofile.close(), but this is also implicitly done by the destructor [p639] so putting a scoping block around line 7-9 will work.

If you do that, then you will get 'This' printed out. Why? Because the ≫ will read tokens delimited by whitespace. If you want to read a whole line you might consider:

```
#include <fstream>
#include <iostream>
#include <string>

int main() {

    std::string ostr = "This is a test of writing and reading from files";
    std::ofstream ofile("myfile.tmp");
    ofile << ostr;
    ofile.close();

    std::string istr;
    std::ifstream ifile("myfile.tmp");
    std::getline(ifile,istr);

    std::cout << istr;
}
```

which might give:

```
g++ -Wall scratch.cpp && ./a.out
This is a test of writing and reading from files
```

```
 1 #include <iostream>
 2 #include <limits>
 3
 4 int main() {
 5
 6     std::cout << std::numeric_limits<char>::digits << std::endl;
 7
 8     std::cout << std::numeric_limits<int>::digits << std::endl;
 9     std::cout << std::numeric_limits<int>::max() << std::endl;
10     std::cout << std::numeric_limits<int>::min() << std::endl;
11 }
```

what might happen if you try to compile, link and run this program?

§x.x, title

I get:

```
g++ -Wall scratch.cpp && ./a.out
7
31
2147483647
-2147483648
```

or:

```
g++ -funsigned-char -Wall scratch.cpp && ./a.out
8
31
2147483647
-2147483648
```

```
 1 #include <iostream>
 2 #include <valarray>
 3
 4 double f(double d) {
 5     return d + 1;
 6 }
 7
 8 int main() {
 9     const double a[] = {1.23, -4.54, 0.48, -1};
10     const double b[] = {1, 0, 0, 1};
11
12     std::valarray<double> va(a,4);
13     std::valarray<double> vb(b,4);
14     std::valarray<double> vc = va * vb;
15     vc *= 2;
16     std::valarray<double> vd = vc.apply(f);
17
18     for (size_t i=-0; i<vd.size(); i++)
19         std::cout << vd[i] << " ";
20 }
```

what might happen if you try to compile, link and run this program?

§x.x, title

I get:

```
g++ -Wall scratch.cpp && ./a.out
3.46 1 1 -1
```

[p662] valarray - designed specifically for speed of the usual numeric vector operations.